PATENT ABSTRACTS OF JAPAN

(11)Publication number : 03-218534
(43)Date of publication of application : 26.09.1991

(51)Int.Cl. G06F 9/46
G06F 15/16

(21)Application number : 02-183942
(22)Date of filing : 13.07.1990

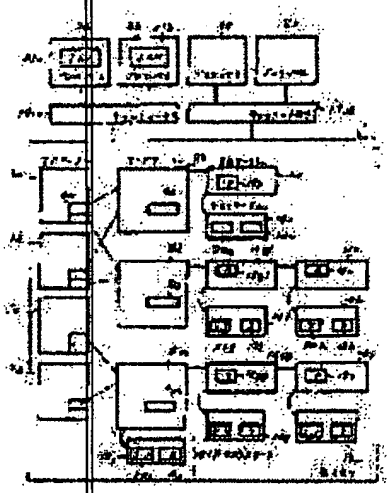
(71)Applicant : HITACHI LTD
(72)Inventor : NEGISHI KAZUYOSHI
FUJII TETSUHIKO
YONEDA SHIGERU
SATO KAZUHIRO
HARADA AKIRA

(30)Priority

Priority number	Priority date	Priority country
01185293	17.07.1989	JP
01304236	21.11.1989	JP

(54) TASK SCHEDULE SYSTEM

(57)Abstract:
PURPOSE: To efficiently use hardware corresponding to a processors by providing virtual processor and permitting the processors to correspond to tasks when hardware that can be used only from a specified processor exists.
CONSTITUTION: When hardware 19ab which can be used only from the specified processor or the groups 2a and 2b exists, the virtual processors are provided, the specified processors 2a and 2b and the tasks are caused to correspond to the virtual processors and the queue of the tasks waiting for execution is provided corresponding to the virtual processors. The specified processors 2a and 2b take out the tasks from the queue of the corresponding virtual processors and start the execution of the tasks. Thus, the possibility of re-use when data which is executed previous time remains on a cache memory 19ab improves, for example, and task search overhead is reduced since respective processors 2a and 2b search only the tasks which can be executed.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

⑲日本国特許庁(JP) ⑪特許出願公開

⑫公開特許公報(A) 平3-218534

⑤Int.Cl.⁵	識別記号	庁内整理番号	⑬公開 平成3年(1991)9月26日
G 06 F 9/46	360 B	8945-5B	
15/16	430	6945-5B	

審査請求 未請求 請求項の数 17 (全16頁)

⑭発明の名称 タスクスケジュール方式

⑳特 願 平2-183942
㉒出 願 平2(1990)7月13日
優先権主張 ㉜平1(1989)7月17日㉝日本(JP)㉞特願 平1-185293

⑰発明者 根岸 和義 神奈川県川崎市麻生区王禅寺1088番地 株式会社日立製作所システム開発研究所内
⑰発明者 藤井 哲彦 神奈川県川崎市麻生区王禅寺1088番地 株式会社日立製作所システム開発研究所内
⑰発明者 米田 茂 神奈川県川崎市麻生区王禅寺1088番地 株式会社日立製作所システム開発研究所内
⑪出願人 株式会社日立製作所 東京都千代田区神田駿河台4丁目6番地
⑭代理人 弁理士 小川 勝男 外1名

最終頁に続く

明 細 書

1．発明の名称

タスクスケジュール方式

2．特許請求の範囲

1．複数のプロセッサが主メモリを共用し、複数のタスクを並行して実行する高多重密結合プロセッサシステムにおいて、特定のプロセッサまたはそのグループのみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、前記特定のプロセッサおよびタスクを前記仮想的なプロセッサに対応付けるとともに、該仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前記特定のプロセッサは、対応する前記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始することを特徴とするタスクスケジュール方式。

2．前記特定のプロセッサまたはそのグループのみから利用可能なハードウェアを使用するタスクを、当該ハードウェアに対応する前記仮想的なプロセッサに対応付けて処理させることを特徴とする請求項1記載のタスクスケジュール方式。

3．前記複数のプロセッサと主メモリとの間に、複数段階のキャッシュメモリを有する場合、ある段階のキャッシュメモリを共用するプロセッサにより前記仮想的なプロセッサを構成し、各プロセッサは、対応する前記仮想的なプロセッサに実行可能なタスクが存在しない場合、他の段階のキャッシュメモリを共用するプロセッサに対応する、前記プロセッサが対応付けらている仮想的なプロセッサ以外の仮想的なプロセッサの処理待ちタスクを探して処理することを特徴とする請求項1記載のタスクスケジュール方式。

4．前記仮想的なプロセッサと前記特定のプロセッサとの対応を、負荷の状態により定期的に変更することを特徴とする請求項1記載のタスクスケジュール方式。

5．タスクに対して対応する前記仮想的なプロセッサの候補を指定する手段を設けて、当初候補

となっている前記仮想的なプロセッサのいずれかでタスクの実行を開始し、以後、同一の前記仮想的なプロセッサに対応付けて処理を行うことを特徴とする請求項１記載のタスクスケジュール方式。

６．各プロセッサ毎に、対応する前記仮想的なプロセッサの処理順序を指定する手段を設けたことを特徴とする請求項１記載のタスクスケジュール方式。

７．高速応答を要求されるタスクに対応する前記仮想的なプロセッサを設けて、該仮想的なプロセッサに対応するプロセッサは、当該仮想的なプロセッサによる処理を優先することを特徴とする請求項１記載のタスクスケジュール方式。

８．前記特定のプロセッサおよび前記仮想的なプロセッサに対応するテーブルを設けて、前記特定のプロセッサに対応するテーブルは対応する前記仮想的なプロセッサへのポインタのリストを有し、前記仮想的なプロセッサからタスクの制御ブロックをチェインする構造により、実行タスクのキューを設けるかわりに、各タスク毎に対応する仮想的なプロセッサを表示したタスクテーブルを設け、前記特定なプロセッサは、各タスクのタスクテーブルにより自プロセッサに対応する仮想的なプロセッサに対応付けられたタスクを選択し、実行を開始することを特徴とする請求項１記載のタスクスケジュール方式。

13．前記仮想的なプロセッサは、全てのプロセッサで処理されるタスクに対応する仮想的なプロセッサであることを特徴とする請求項１２記載のタスクスケジュール方式。

14．前記ハードウェアはキャッシュメモリであることを特徴とする請求項１２記載のタスクスケジュール方式。

15．前記複数のプロセッサは主メモリを共用するとともに、複数のアドレス空間を有し、少なくとも前記アドレス空間の一つで複数のタスクを並行して実行することを特徴とする請求項１２記載のタスクスケジュール方式。

16．前記アドレス空間に対応する空間テーブルを設け、各仮想的なプロセッサに対応する仕事が該アドレス空間にあることを表示する手段を該空間テーブルに設け、タスクスケジュール時に該表示手段を参照することによりタスクを実行することを特徴とする請求項１５記載のタスクスケジュール方式。

17．前記特定の仮想的なプロセッサに対応付けられたプロセッサまたはそのグループが故障した場合、それ以外の故障していないプロセッサまたはそのグループを、新たに前記仮想的なプロセッサに対応付けることを特徴とする請求項１４に記載のタスクスケジュール方式。

３．発明の詳細な説明

〔産業上の利用分野〕

本発明は計算機システムにおけるタスクスケジュール方式に関し、特に複数のプロセッが主メモリを共用し、複数の処理（タスク）を並行して実行する高多重密結合プロセッサシステムにおける高性能タスクスケジュール方式に関する。

〔従来の技術〕

可能なタスクのキューを構成することを特徴とする請求項１記載のタスクスケジュール方式。

９．前記プロセッサシステムの操作を行うための端末から入力された指示により、前記特定のプロセッサと前記仮想的なプロセッサとの関係および前記仮想的なプロセッサとタスクとの関係を前記端末に出力することを特徴とする請求項１記載のタスクスケジュール方式。

10．前記複数のプロセッサが主メモリを共用するとともに、複数のアドレス空間を有し、少なくとも前記各アドレス空間の一つで、複数のタスクを並行して実行することを特徴とする請求項１～９のいずれかに記載のタスクスケジュール方式。

11．複数のタスクが前記アドレス空間に対応付けられて処理される場合に、前記タスクの代りにタスクに対応する前記アドレス空間を、前記仮想的なプロセッサに対応付けることを特徴とする請求項１記載のタスクスケジュール方式。

12．前記仮想的なプロセッサに対応して実行待ち

従来、タスクスケジュール方式については、先入れ先出し方式、プライオリティによる方式、プロセッサ使用時間の長いタスクよりＩ／Ｏの多いタスクを優先する方式、処理時間の目標値を守るように処理順序を変える方式等があった。これらの方式では、タスクの切替えオーバヘッドは、考慮されていなかった。

また、タスクの切替えオーバヘッドを考慮したタスクスケジュール方式としては、例えば、特開昭63-113738号 公報に開示された方式が知られている。しかし、この方式においても、プロセッサ間で共用されているキャッシュメモリ等のハードウェアを意識したタスクスケジュール方式は配慮されていない。

特定のタスクを特定のハードウェアリソースが利用可能なプロセッサ（またはそのグループ）でのみ実行させるタスクスケジュール方式として、特開昭６２－７５７３９公報（米国特許4809157に対応）に開示された方式が知られている。この方式では、タスク毎に設けられたタスクテーブル内に当該タスクを実行可能なプロセッサをビットマップとして設定する。タスクスケジュール時に当該ビットマップをチェックし、許可されたタスクのみをスケジュールすることにより、上記のプロセッサ限定実行を実現している。

【発明が解決しようとする課題】

前記各従来技術は、タスクを実行中のプロセッサが、その時点での最も優先度の高いタスクを実行するように優先順位により配列された実行可能なタスクのキューをサーチし、タスクを切替えながら動作する。従って、前回タスクを実行したときのデータがキャッシュメモリに残っており、これを再利用することにより、キャッシュミスを低減し、ＭＩＰＳを向上させることが可能な場合について配慮がなされておらず、当該キャッシュメモリを利用できないプロセッサにおいて、タスクの実行を開始してしまい、このため、余分なキャッシュミスが発生するという問題があった。

また、特定のプロセッサでのみ実行可能なタスクのスケジュールオーバヘッドの削減に関する配慮がなされておらず、プロセッサを限定して処理を依頼しようとしても、すべてのプロセッサが実行可能なタスクのキュー上のすべてのタスクをサーチして実行するタスクを選択しているため、指定外のプロセッサがキュー上で当該タスクの実行可能性のチェックをして、スキップするオーバヘッドが大きいという問題もあった。以下、これについて具体的に説明する。

第６図は、従来のタスクスケジュール方式における、プロセッサ間で共用されているキャッシュメモリ等のハードウェアを意識したシステム構成例を示す図である。このシステムにおいては、主メモリ１ａを共用するプロセッサ２ａ，２ｂ，２ｃおよび２ｄは、各々独立にタスクを処理することができる。プロセッサ２ａと２ｂ，２ｃと２ｄはそれぞれ、キャッシュメモリ１９ａｂ，１９ｃｄを共用する。また、システムテーブル５およびスケジュールロックエントリ７を主メモリ１ａ上に持つ。

主メモリ１ａ上には、タスク情報を管理するための領域（タスクテーブル）１０ａ，１０ｆ，１０ｇ，１０ｈ，１０ｉが、タスク対応に設けられており、同様に、タスク間の処理の独立性を高めるために使用される多重アドレス空間を管理するための領域（空間テーブル）１４ｐ，１４ｑ，１４ｒ，１４ｓが空間対応に設けられている。この空間テーブルのうち、実行可能なタスクテーブルをチェインしているものは、空間の優先順位１５ｐ，１５ｑ，１５ｓに従って、システムテーブル５からチェインされている。

ここでは、タスクテーブル１０ｈに対応するタスク（以下、単に「タスク１０ｈ」等という）は、プロセッサ２ａ，２ｂにのみ実装されている内蔵アレープロセッサ（以下、「ＩＡＰ」という）を使用するため、プロセッサ２ａ，２ｂでのみ実行可能であることを示す印（フラグ）が付けられている。

第８図において、プロセッサ２ａがタスクスケジュールを行う場合、先頭の空間テーブル１４ｐの先頭のタスク１０ａが選択され実行される。こ

のとき、タスクテーブル１０ａおよび実行可能なタスクがなくなった空間テーブル１０ｅおよび実行可能なタスクがなくなった空間テーブル１４ｐは、それぞれのキューから外されるのが普通である。

更に、プロセッサ２ｂ，２ｃがタスクスケジュールを行うと、それぞれ、タスク１０ｆ、１０ｇが選択され、実行される。次に、プロセッサ２ｄがタスクスケジュールを行うと、選択されるタスクは順序から言えばタスク１０ｈであるが、前述の如くタスク１０ｈには、プロセッサ２ａ，２ｂでのみ実行可能であることを示す印（フラグ）が付けられているので、これを検知してスキップし、次のタスク１０ｉが選択され実行される。なお、タスクテーブル、空間テーブルのサーチと操作を行う際のプロセッサ間の排他を行うため、タスクスケジュールの間、スケジュールロックエントリ７がロックされる。

前述の如く、上記技術によれば、特定のプロセッサでのみ実行可能なタスクのスケジュールオーバヘッドの削減に関する配慮がなされていないため、プロセッサを限定して処理を依頼しようとしても、すべてのプロセッサが実行可能なタスクのキュー上のすべてのタスクをサーチして実行するタスクを選択しているため、指定外のプロセッサがキュー上で当該タスクの実行可能性のチェックをすることにより、スキップするオーバヘッドが大きいという問題があった。

また、タスクに対応するタスクテーブル毎に実行可能なプロセッサとの対応関係を設定しており、この対応関係を変更する必要の生じた場合、タスク数が多いと、変更オーバヘッドが大きいという問題があった。

本発明は上記事情に鑑みてなされたもので、その目的とするところは、従来の技術における上述の如き問題を解消し、上述のプロセッサ対応のハードウェアの効率的な利用を図り、更に、オーバヘッドの少ないプロセッサ限定タスクスケジュールを可能とするタスクスケジュール方式を提供することにある。

〔課題を解決するための手段〕

このような目的を達成するために、本発明は、複数のプロセッサが主メモリを共用し、複数の処理（タスク）を並行して実行する高多重密結合プロセッサシステムにおいて、特定のプロセッサ（またはそのグループ）のみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、前述の特定のプロセッサおよびタスクを前記仮想的なプロセッサに対応付けるとともに、下記のいずれかを実行することを特徴とする。

（１）前記仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前述のプロセッサは、対応する前記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始する。

（２）各タスク毎に対応する仮想的なプロセッサを表示したタスクテーブルを設け、前記プロセッサは、各タスクのタスクテーブルにより自プロセッサに対応する仮想的なプロセッサに対応付けられたタスクを選択し、実行を開始する。

〔作用〕

本発明に係るタスクスケジュール方式においては、特定のプロセッサ（またはそのグループ）のみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、上述の特定のプロセッサおよびタスクを上記仮想的なプロセッサに対応付けるとともに上記仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前述の特定のプロセッサは、対応する上記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始するようにしたので、例えば、キャッシュメモリ上に前回実行時のデータが残っている場合に、その再利用の可能性を向上させることができる。更に、各プロセッサは、当該プロセッサで処理不可能なタスクのサーチを行わず、実行可能なタスクのサーチのみを行うことにより、タスクサーチオーバヘッドの削減を実現している。

また、別のタスクサーチオーバヘッド削減策として、タスクスケジュール時に、仮想的なプロセ

ッサに対応する仕事がアドレス空間にある場合にのみアドレス空間内のタスクのサーチを実行し、空間レベルでのスキップを行うことにより、各プロセッサは当該プロセッサで処理不可能なタスクのサーチを最小限とする方策を実現している。

また、プロセッサあるいはハードウェアリソースの故障等によりタスクとプロセッサの対応付けを変更する必要の生じた場合、プロセッサと仮想的なプロセッサとの対応付けを変更すれば良い。一般的に、タスク数よりはタスクを対応付ける対象となるハードウェアリソース数は少ない。従って、対応関係の変更はタスク対応のテーブル内にプロセッサとの対応付けを保持している場合と比較してより少ないオーバヘッドで実現可能である。

【実施例】

以下、本発明の一実施例を図面を用いて具体的に説明する。

第２図は、本発明の一実施例に係るタスクスケジュール方式におけるシステム構成を示す図である。主メモリ１を共用するプロセッサ２ａ，２ｂ，２ｃおよび２ｄは、各々独立にタスクを処理することができる。プロセッサ２ａと２ｂ，２ｃと２ｄは、それぞれキャッシュメモリ１９ａｂ，１９ｃｄを共用し、また、システムテーブル５およびスケジュールロックエントリ７を主メモリ１上に持つ点は先に示したシステムと同様である。

上記主メモリ上には、プロセッサ２ａ，２ｂ，２ｃおよび２ｄにそれぞれ対応して、ＩＰテーブル３ａ，３ｂ，３ｃおよび３ｄが設けられている。また、重複を許すプロセッサグループに対応する仮想プロセッサ（ＶＩＰ）に対応するテーブル（以下、「ＶＩＰテーブル」という）８ｋ，８ℓ，８ｍが設けられている。なお、本実施例においては、ＶＩＰテーブル８ℓはプロセッサ２ａ，２ｂにのみ実装されている前記ＩＡＰおよびキャッシュメモリ１９ａｂに対応し、ＶＩＰテーブル８ｍはキャッシュメモリ１９ｃｄに対応している。

また、主メモリ１上には、タスク情報を管理するための領域（以下、「タスクテーブル」という）１０ｅ，１０ｆ，１０ｇ，１０ｈ，１０ｉが、タスク対応に設けられている。タスクテーブル１０ｅ～１０ｉには、それぞれ、ＶＩＰ番号１２ｅ～１２ｉおよびＶＩＰ候補番号２０ｅ～２０ｉを予め設定する。更に、タスクテーブル１０ｅ～１０ｉをアドレス空間対応にＶＩＰテーブルにチェインするための制御ブロックとして、ＳＱ（スケジュールキュー）テーブル１６ｐ～１６ｓを用意する。なお、各アドレス空間の全タスクが、各々、同一のＶＩＰテーブルからチェインする方式も可能である。

なお、ここでも、タスク１０ｅは、優先タスクとして、ＶＩＰ候補番号２０ｅおよびＶＩＰ番号１２ｅには１のみを設定する。同様に、ＩＡＰを使用するタスク１０ｈに対しては、２のみを設定する。ＶＩＰテーブル３ａ，３ｂのＶＩＰリスト４ａ，４ｂには、ＶＩＰテーブル８ｋ，８ℓへのポインタをこの順序で設定する。また、ＩＰテーブル３ｃ，３ｄには、ＶＩＰテーブル８ｍへのポインタを設定する。第３図は本実施例の方式におけるタスクテーブルと空間テーブル、ＳＱテーブルの関係を示す図である。タスクテーブルから空間テーブルを辿ることができる。また、空間テーブルから、ＶＩＰ番号毎に対応するＳＱテーブルを辿ることができる。

本実施例における、データ等が揃って、タスクが実行可能になった時の処理の流れを第７図に示す。まず、タスクテーブルのＶＩＰ番号（前述の仮想プロセッサに付加されている一意番号）から、対応するＶＩＰを求め（ステップ７１）、ステップ７２で、ＶＩＰテーブルのＶＩＰタスクスタックにＣＳ（Compare and Swap）命令により挿入する。第２図に、ＶＩＰテーブル８ｍにタスクテーブル１０ｊをスタックした例が示されている。

本実施例におけるプロセッサによるタスクスケジュールの処理の流れを第１図に示す。まず、ステップ６０１で、自プロセッサに対するＩＰテーブルを見つける。そして、ＩＰテーブルのＶＩＰリストから最初のＶＩＰを求める（ステップ６０２）。次に、ＶＩＰテーブル中のロックエントリをロックし（ステップ６０３）、ＶＩＰタスクスタックに

タスクスタテーブルがスタックされているかチェックする(ステップ６０４)。スタックされている場合は、ＶＩＰタスクスタックから全タスクテーブルをＣＳ命令により外す(ステップ６０５)。タスクテーブルを１つとり(ステップ６０６)、ＴＳＫＥＮＱ(Task Enqueue)ルーチンをコールして、タスクテーブルをＶＩＰのキューに入れる(ステップ６０７)。未処理のタスクテーブルがまだあるなら(ステップ６０８)、更に、ＴＳＫＥＮＱを繰り返す。

ＶＩＰテーブルにＳＱテーブルがチェインされていない場合（ステップ６０９）には、ＶＩＰテーブルのロックエントリのロックを外す（ステップ６１０）。ＩＰテーブルのＶＩＰリストに、次のＶＩＰテーブルが含まれているならばプロセッサはＷＡＩＴに入る。ＶＩＰにＳＱテーブルがチェインされている場合、先頭のＳＱテーブルを求め(ステップ６１２)、ＳＱテーブルにチェインされているタスクテーブルの先頭のものを求める(ステップ６１３)、ＴＳＫＤＥＱ(Task Dequeue)ルーチンをコールして、タスクテーブルをＳＱテーブルのチェインから外す（ステップ６１４）、ＶＩＰテーブルのロックエントリのロックを外し（ステップ６１５）、対応するタスクの実行を開始する。

第９図は、上記ステップ６０７に示したＴＳＫＥＮＱルーチンの処理の流れである。まず、第３図に例を示す如く、タスクテーブル１０ｊからＶＩＰ番号１２ｊを求め（ステップ９１）、更に、空間テーブル１４ｐから、ＶＩＰ番号１２ｊに対するＳＱテーブル１６ｐｇを、ＳＱテーブルリスト１８ｐを使用して求める（ステップ９２）。次に、タスクテーブルを、ＳＱテーブルからのタスクテーブルのキューの末尾に挿入する（ステップ９３）。ＳＱテーブルが、まだＶＩＰテーブルからチェインされていない場合には(ステップ９４)、ＳＱテーブルを、ＶＩＰテーブルのチェインに優先順位２０ｐｚを考慮して挿入する（ステップ９５）。

第１０図は、前述のステップ６１４に示したＴＳＫＤＥＱルーチンの処理の流れを示すものである。まず、タスクテーブルをＳＱテーブルのチェインから外す（ステップ１０１）。ＳＱテーブルにタスクテーブルがなくなった場合（ステップ１０２）には、ＳＱテーブルを、ＶＩＰテーブルのチェインから外す（ステップ４１０３）。

上記実施例によれば、実行中断したタスクが、再度スケジュールされる際、同一のキャッシュメモリを有するプロセッサで実行される可能性が高くなり、また、直接このタスクを実行開始できるプロセッサ以外は、当該タスクをサーチせず、タスクスケジュールにおけるサーチオーバヘッドを削減する効果もある。

次に、本発明の他の実施例を、第４図に示すテーブル関連図および第８図の処理の流れ図により説明する。第４図に示す如く、例えば、ＩＰテーブル３ｃには高負荷時のＶＩＰリスト２１ｃ、および低負荷時のＶＩＰリスト２２ｃが予め設定されている。負荷変動の監視処理は、第８図に示す如く、定期的に起動され、別に設定されている手段により測定された前回監視処理を行ってからのＣＰＵ使用時間を使用して、ＣＰＵ使用率を求める（ステップ８１）。ＣＰＵ使用率が予め与えられた限界値より大きい場合（ステップ８２）、全ＩＰテーブル中のＶＩＰを高負荷時のＶＩＰに従って変更する（ステップ８３）。ＩＰテーブル３ｃにおいては、ＶＩＰリスト４ｃとしてＶＩＰテーブル８ｍのみを設定する。第４図の設定内容に対する高負荷時のＩＰテーブルとＶＩＰテーブルの対応関係は、第２図の通りである。また、ＣＰＵ使用率が限界値より低い場合、全ＩＰテーブル中のＶＩＰリストを低負荷時のＶＩＰリストに従って変更する（ステップ８４）。ＩＰテーブル３ｃにおいては、ＶＩＰリスト４ｃとして、ＶＩＰテーブル８ｋおよびＶＩＰテーブル８ｍを設定する。第４図は、低負荷時のＩＰテーブルとＶＩＰテーブルの対応関係を示している。

上記実施例によれば、ＶＩＰとプロセッサとの対応を、プロセッサの負荷状態により変更することができ、処理の効率向上が可能になるという効

果がある。

次に、プロセッサシステムの操作を行うための端末から入力された指示により、上記プロセッサとＶＩＰとの関係およびＶＩＰとタスクとの関係を端末に出力する例を示す。第４図のＶＩＰテーブル８ｋに示す如く、テーブル中には、当該ＶＩＰに対応するタスクの数２４ｋと、変更不可タスクの数２５ｋが設定されている。このカウントは、タスクとＶＩＰの対応を変更する都度、更新する。例えば、第２図に示したタスクテーブル１０ｆは、ＶＩＰ候補番号２０ｆが２および３の複数となっており、変更可能である。これに対して、タスクテーブル１０ｈは、ＶＩＰ候補番号２０ｈが２のみであり、変更不可タスクである。従って、ＶＩＰテーブル８ｇのタスク数２４ｇは２であり、変更不可タスク数は１である。

第５図に示したＶＩＰとタスクの関連表示は、このＶＩＰテーブル中のタスク数および変更不可タスク数による。また、ＩＰとＶＩＰの関連表示は、ＩＰテーブル中のＶＩＰリスト、高負荷時のＶＩＰリストおよび低負荷時のＶＩＰリストを使用して行う。

上記実施例によれば、必要に応じて、プロセッサシステムの操作を行うための端末から指示を行うことにより、プロセッサとＶＩＰとの関係およびＶＩＰとタスクとの関係を端末に出力させ、状態を確認することができる。

次に、本発明の更に別の実施例を示す。本実施例は、前述の実施例のタスクテーブル操作および排他制御方式を変更し、ロックの競合確率を低下させるものである。

第１１図は、本実施例の方式におけるシステム構成図であり、以下の点を除いては、第２図に示した実施例と同様である。

ＶＩＰテーブル８ｋ，８ｇ，８ｍの各々に対応して、下記のＶＩＰＭＳＧ（ＶＩＰ Message）スタックを設ける。第１１図においては、ＶＩＰテーブル８ｍのＶＩＰＭＳＧスタックにＭＳＧ（Message）テーブル１１１ｖが入れられている。ＳＱテーブル１６ｐ～１６ｓには、ロックエントリ１１０ｐ～１１０ｓを設定する。また、各ＳＱテーブル対応に、ＳＱＭＳＧ（ＳＱ Message）スタックを用意する。第１１図では、ＳＱテーブル１６ｑｇのＳＱＭＳＧスタックにＭＳＧテーブル１１１ｗが入れられている。

更に、システムテーブル５からチェインされたＭＳＧテーブルのチェインが未使用ＭＳＧプールとして追加されている。第１１図では、未使用ＭＳＧプールに、ＭＳＧテーブル１１１ｓ，１１１ｔ，１１１ｕが用意されている。第１１図ではタスクテーブル中のＶＩＰ候補番号を省略しているが、本実施例でも同様に設定されているものとする。また、本実施例におけるタスクテーブルと空間テーブル、ＳＱテーブルの関係は、先の実施例の場合と同様である（第８図参照）。

本実施例における、タスクが実行可能となったときの処理の流れを、第１３図に示す。まず、未使用ＭＳＧプールから１つ、ＭＳＧテーブルを外す（ステップ１３１）。このＭＳＧテーブルにタスクテーブルのアドレスを設定する（ステップ１３２）。タスクテーブルのＶＩＰ番号から、対応するＶＩＰを求め（ステップ１３３）、ＶＩＰテーブルのＭＳＧスタックにＣＳ命令により、前記ＭＳＧテーブルを挿入する（ステップ１３４）。第１１図には、ＶＩＰテーブル８ｍにＭＳＧテーブル１１１ｖをスタックした例を示した。

本実施例におけるプロセッサによるタスクスケジュールの処理の流れを、第１２図に示す。まず、自プロセッサに対するＩＰテーブルを見つける（ステップ１２０１）。ＩＰテーブルのＶＩＰから、最初のＶＩＰを求める（ステップ１２０２）。ステップ１２０３では、ＶＩＰテーブル中のロックエントリをロックし、ＶＩＰＭＳＧスタックにＭＳＧテーブルがスタックされているか否かをチェックする（ステップ１２０４）。スタックされている場合は、ＶＩＰＭＳＧスタックから、全ＭＳＧテーブルをＣＳ命令により外す（ステップ１２０５）。

ＭＳＧテーブルを１つとり（ステップ１２０６）、ＳＱＥＮＱ（ＳＱ Enqueue）ルーチンをコールし

て、ＭＳＧテーブルをＳＱテーブルのＳＱＭＳＧスタックに入れるとともに、ＳＱテーブルをＶＩＰテーブルのキューに入れる(ステップ１２０７)。未処理のＭＳＧテーブルがまだあるなら(ステップ１２０８)、更にＳＱＥＮＱを繰り返す。ＶＩＰテーブルに、ＳＱテーブルがチェインしていない場合(ステップ１２０９)には、ＶＩＰテーブルのロックエントリのロックを外す(ステップ１２１０)。なお、ＩＰテーブルのＶＩＰリストに、次のＶＩＰテーブルが含まれているなら(ステップ１２１１)、当該ＶＩＰテーブルに対して、上述の処理を繰り返す。ＶＩＰテーブルが含まれていない場合は、プロセッサはＷＡＩＴに入る。

ＶＩＰテーブルにＳＱテーブルがチェインされている場合、操作対象としてチェイン先頭のＳＱテーブルを求め(ステップ１２１２)、ＳＱＭＳＧスタックにＭＳＧテーブルあり、または、ＳＱテーブルにタスクテーブルのチェインあり、をチェックする(ステップ１２１３)。どちらかが有の場合には、ＶＩＰテーブルのロックエントリのロックを外し(ステップ１２１４)、ＳＱテーブルのロックエントリのロックを確保し(ステップ１２１５)、ＡＳＳＣＨ(Address Space Schedule)ルーチンをコールして、空間に関するスケジュールを行う(ステップ１２１６)。

次に、ＳＱテーブルのロックエントリのロックを外し(ステップ１２１７)、実行開始するタスクがみつかったか否かをチェックする(ステップ１２１８)。見つかったときは対応するタスクの実行を開始する。見つからないときはＶＩＰテーブルのロックエントリを再度ロックし(ステップ１２１９)、ＳＱテーブルが現在ＶＩＰのＳＱテーブルチェインに入っているか否かをチェックする。(ステップ１２２０)。ＳＱテーブルがチェインから外されている場合には、ステップ１２０９へ戻り、他のＳＱテーブルをさがす。また、ＳＱテーブルがあるときは、ＳＱテーブルをＶＩＰのＳＱテーブルチェインから外す(ステップ１２２１)。更に、次のＳＱテーブルを求めて、あればその処理を行い(ステップ１２１０)、なければ、ＶＩＰのロックエントリのロックを外して(ステップ１２１３)、次のＶＩＰの処理へ移る。

第１４図は、ＳＱＥＮＱルーチンの処理の流れを示すものである。まず、ＭＳＧテーブルからタスクテーブルを辿り、そこからＶＩＰ番号を求めて(ステップ１４１)、更に、空間テーブルからＶＩＰ番号に対するＳＱテーブルを、ＳＱテーブルリストを使って求める(ステップ１４２)、ＭＳＧテーブルをＳＱテーブルからのＳＱＭＳＧスタックに挿入する(ステップ１４３)。なお、ＳＱテーブルがまだＶＩＰテーブルからチェインされていない場合(ステップ１４４)には、ＳＱテーブルを、ＶＩＰテーブルのチェインに、優先順位を考慮して挿入する(ステップ１４５)。

第１５図に、空間内のタスク処理を行うＡＳＳＣＨルーチンの処理の流れを示す。まず、ＳＱＭＳＧスタックに、ＭＳＧテーブルがあるか否かをチェックする(ステップ１５１)。ＭＳＧテーブルがあるとき、すべてのＭＳＧテーブルをＳＱＭＳＧスタックから外す(ステップ１５２)、ＭＳＧテーブルを１つとり(ステップ１５３)、当該ＭＳＧテーブルに対応するタスクテーブルが、既にＳＱテーブルからチェインされているか否かをチェックする(ステップ１５４)。チェインされていないときは、タスクテーブルをＳＱテーブルのタスクテーブルキューの末尾に挿入する(ステップ１５５)。次に、ＭＳＧテーブルを、未使用ＭＳＧプールに戻す(ステップ１５６)、これを、ＳＱＭＳＧスタックから外したＭＳＧテーブルの、すべてに関して行う(ステップ１５７)。最後に、実行可能なタスクのタスクテーブルが、ＳＱテーブルからチェインされているか否かをチェックし(ステップ１５８)、ステップ１５９では、その先頭のタスクテーブルを、実行開始のためＳＱテーブルのタスクテーブルキューから外す。

次に、本発明の更に他の実施例を、第１６図に示すシステム構成図を基に説明する。本実施例では、キャッシュメモリが二段階になっており、図に示す如く、第２図に示した実施例に対して、プ

ロセッサ対応の第二のキャッシュメモリ１６１ａ～１６１ｄが追加されている。また、ＶＩＰテーブル８ａ～８ｄは各プロセッサ２ａ～２ｄに対応して設定されている。各プロセッサとＶＩＰとの対応関係は、ＩＰテーブル中のＶＩＰリストにより、以下の如く設定されている。

各プロセッサは、まず、対応するＶＩＰをチェックする。次に、当該ＩＰとキャッシュメモリを共用するプロセッサに対応するＶＩＰを、チェックする。例えば、プロセッサ２ａは、まず、ＶＩＰテーブル８ａをチェックする。これにより、キャッシュメモリ１９ａｂおよび第二のキャッシュメモリ１６１ａのデータを再利用する確率を高める。このようなタスクが、ＶＩＰテーブル８ａに存在しない場合、ＶＩＰテーブル８ｂをチェックすることにより、キャッシュメモリ１９ａｂのデータを再利用する確率を高める。その他の動作および処理の流れは、前述の実施例と同様である。

第１７図は、本発明の更に他の実施例におけるシステム構成図である。

主メモリ１を共用するプロセッサ２ａ，２ｂ，２ｃ，２ｄは、おのおの独立にタスクを処理することができる。プロセッサ２ａと２ｂ，２ｃと２ｄはそれぞれキャッシュメモリ１９ａｂおよび１９ｃｄを共用する。システムテーブル５およびスケジュールロックエントリ７を主メモリ上に持つ点は前述した従来のシステムと同様である。

プロセッサ２ａ，２ｂ，２ｃ，２ｄに対応してＩＰテーブル３ａ，３ｂ，３ｃ，３ｄを設ける。また、ＩＰテーブルには、ＶＩＰとの対応関係をビットにより表現したサービス対象ＶＩＰビットマップ３１ａ，３１ｂ，３１ｃ，３１ｄが設けられている。ビットマップのビット０，１，２は、三種のＶＩＰ（ＶＩＰ０，ＶＩＰ１，ＶＩＰ２）と対応を各々表している。ビットが１の時、そのＩＰはＶＩＰに対応付けられている。

本発明の実施例では、ＶＩＰ０は全ＩＰで処理を行うタスクに、ＶＩＰ１はキャッシュメモリ１９ａｂに対応し、ＶＩＰ２はキャッシュメモリ１９ｃｄに対応する。タスクテーブル１０ｆには仕事有ビットマップ１１４ｆ（ＶＩＰ１に対するタスクであることを表わす）をあらかじめ設定する。タスクテーブル１０ｆを空間対応にシステムテーブル５にチェインするための制御ブロックとして、空間テーブル１４ｑを用意する。他のタスクテーブルも同様である。タスクテーブル１０ｑに対しては仕事有ビットマップ１１４ｑを設定する。ＩＰテーブル３ａ，３ｂのサービス対象ＶＩＰビットマップ３１ａ，３ｂのサービス対象ＶＩＰとの対応を表わすビットを設定する。また、ＩＰテーブル３ｃ，３ｄには同様にしてＶＩＰ０，ＶＩＰ２との対応を設定する。

各空間テーブルには、ロックエントリ１１０ｑ、仕事有ＶＩＰビットマップ１１２ｑ（左から順にＶＩＰ０，１，２に対応）、ＶＩＰ対応カウンタ１１３ｑ（上から順にＶＩＰ０，１，２に対応）が設けられている。

第１８図は、本発明のタスクを実行可能な状態とした時の処理の流れを示す。まず、そのタスクに対応する空間テーブルを求める（ステップ１７０１）。次に、その空間テーブルのロックエントリをロックし（ステップ１７０２）、タスクを空間テーブルのタスクキューに入れる（ステップ１７０３）。タスクテーブルの仕事有ビットマップにおいてオンとなっているビットに対応するＶＩＰに対して、つぎの処理を行う（ステップ１７０４，１７０７，１７０７，１７０８）。空間テーブルのＶＩＰ対応カウンタをプラスし、空間テーブルの仕事有ＶＩＰビットマップの対応するビットをオンに設定する。空間テーブルのロックエントリのロックを外し（ステップ１７０９）、この空間テーブルが空間テーブルキューに入っているかチェックしなければ、空間テーブルを空間テーブルキューに入れる（ステップ１７１２）。空間テーブルキューの操作の間、スケジュールロックエントリのロックを確保しておく（ステップ１７１１，１７１３）。

本発明におけるプロセッサによるタスクスケジュールの処理の流れを第１９図に示す。まず、自プロセッサに対するＩＰテーブルを見つける（ス

テップ１８０１）．ＩＰテーブルからサービス対象ＶＩＰビットマップを求める（ステップ１８０２）．スケジュールロックエントリをロックし（ステップ１８０３）．空間テーブルキュー先頭の空間テーブルを求める（ステップ１８０４）．

システムテーブルに空間テーブルがチェインしていない場合（ステップ１８０５），スケジュールロックエントリのロックを外し（ステップ１８１１），プロセッサはウェートに入る．空間テーブルがチェインしている場合は，対象となった空間テーブルの仕事有ＶＩＰビットマップと前記サービス対象ＶＩＰビットマップの論理積を求め（ステップ１８０６），結果がオールゼロかチェックする（ステップ１８０７），結果がオールゼロの場合，前記空間テーブルの仕事有ＶＩＰビットマップがオールゼロかをチェックする（ステップ１８０８），これもオールゼロの場合，対象空間テーブルを空間テーブルキューから外す（ステップ１８０９）．いずれの場合も次の空間テーブルを求め（ステップ１８１０）ステップ１８０５へ戻る．

前述の結果がオールゼロでない場合，スケジュールロックエントリのロックを外し（ステップ１８１２），空間テーブルのロックエントリをロックする（ステップ１８１３），空間のタスクキュー先頭のタスクテーブルを求める（ステップ１８１４），タスクテーブルのない場合，空間テーブルのロックエントリのロックを外し（ステップ１８１８），スケジュールロックエントリをロックし（ステップ１８１９），ステップ１８１０へ戻る．タスクテーブルのある場合，タスクテーブルの仕事有ビットマップと前述のサービス対象ＶＩＰビットマップの論理積を求め（ステップ１８１５），結果がオールゼロなら次のタスクテーブルを求め（ステップ１８１７），ステップ１８１５へ戻る．論理積がオールゼロでない場合，タスクキュー（ＴＳＫＤＥＱ）ルーチンをコールして，タスクテーブルを空間テーブルのチェインから外す（ステップ１８２０）．空間テーブルのロックエントリのロックを外し（ステップ１８２１），対応するタスクの実行を開始する．

第２０図にタスクキュー（ＴＳＫＤＥＱ）ルーチンの処理の流れを示す．まず，タスクテーブルを空間テーブルのチェインから外す（ステップ１９０１）．タスクテーブルの仕事有ビットマップでオンとなっているビットに対応するＶＩＰの各々に対して下記の処理を行う（ステップ１９０２，１９０６，１９０７）．空間テーブルのＶＩＰ対応の仕事有カウンタをマイナス１する（ステップ１９０３）．カウンタがゼロになったなら（ステップ１９０４），空間テーブルの仕事有ＶＩＰビットマップの対応するビットをＣＳ（Compare and Swap）命令によりリセットする（ステップ１９０５）．

第２１図は，ＶＩＰがキャッシュメモリのように性能は低下するが他のＶＩＰのもので代替可能なハードウェアリソースに対応して設定されている場合のプロセッサ故障時の処理の流れを示す．第１７図に示すように，例えばＩＰテーブル３ａ，３ｂにはサービス対象ＶＩＰビットマップ３１ａ，３ｂ（ＶＩＰ０およびＶＩＰ１を指定）があらかじめ設定されている．

そして，例えばプロセッサ２ａの故障時にプロセッサ故障時の処理が起動され，第２１図に示すように，故障したプロセッサのサービス対象ＶＩＰ（ここではＶＩＰ０およびＶＩＰ１）を求める（ステップ２００１）．各ＶＩＰに対して下記の処理を行う（ステップ２００８，２００７）．正常に動作中のプロセッサのＩＰテーブルをサーチし（ステップ２００２，２００４，２００５），前記ＶＩＰに対するサービス対象ＶＩＰビットマップがオフのＩＰテーブルを見つけたなら当ビットをオンに変える（ステップ２００３，２００８）．この例では，ＶＩＰ０に対しては，これをサービスしないプロセッサはないため，テーブルの書替は発生しない．しかし，ＶＩＰ１に対してはＩＰテーブル３ｏのサービス対象ＶＩＰビットマップ３１ｏにおいて，ＶＩＰ１に対応するビットが設定され，「１１１」となる．

次にプロセッサ３ｂが故障となった場合，同様

の処理によりＩＰテーブル３ｄのサービス対象ＶＩＰビットマップ３１ｄにおいて、ＶＩＰ１に対応するビットが設定される。これにより、ＶＩＰ１をサービスするプロセッサがなくなることなく、処理を継続することが可能である。

第２２図は、プロセッサシステムの操作を行うための端末から入力された指示により、空間とＶＩＰタスクの関連を端末に表示する例である。第１７図の空間テーブル１４ｑに示すように、テーブル中には当該空間に対応する各ＶＩＰ対応のタスクの数１１３ｑが設定されている。このカウントは第１８図および図２０図の処理フローにより示した通り、空間のタスクキューのタスクを出し入れする都度、更新する。例えば第１７図において、空間テーブル１４ｑはＶＩＰ０に対するタスクが１個、ＶＩＰ１に対するタスクが１個そのタスクキュー中に存在する。これに対して、空間テーブル１４ｒはＶＩＰ１のタスクが１個のみである。第２２図に示す空間とＶＩＰタスクの関連表示は、この空間テーブル中のカウンタによる。この例では、空間テーブル１４ｑは第一行目に、空間テーブル１４ｒは第二行目に表示されている。また、各ＶＩＰに対するタスク数の合計が最終行に表示されている。

なお、前述の各実施例の説明においては、本発明を複数のプロセッサが主メモリを共用するとともに、複数のアドレス空間を有し、少なくとも前記各アドレス空間の一つで、複数の処理（タスク）を並行して実行する如く構成されたシステムに対して適用して例のみを示したが、本発明はこれに限定されるものではなく、単一のアドレス空間のみを有するシステムに対しても、有効に適用可能であることは言うまでもない。

【発明の効果】

以上、詳細に説明した如く、本発明によれば、複数のプロセッサが主メモリを共用し、複数の処理（タスク）を並行して実行する高多重密結合プロセッサシステムにおいて、特定のプロセッサのみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、前述のプロセッサおよびタスクを前記仮想的なプロセッサに対応付けるとともに（１）前記仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前述のプロセッサは、対応する前記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始する、あるいは、（２）タスクおよび空間毎に対応する仮想的なプロセッサのビットマップを設けたことにより、空間レベルでビットがオンとなっている仮想的なプロセッサに対応するプロセッサ以外は当該空間内のタスクをサーチしないことにより、プロセッサ対応のハードウェアの効率的な利用を図り、更に、オーバヘッドの少ないプロセッサ限定タスクスケジュールを可能とするタスクスケジュール方式を実現できるという顕著な効果を奏するものである。

また、仮想的なプロセッサを設けたことにより、タスクプロセッサの対応付けの変更が低いオーバヘッドで実現可能となっている。

４．図面の簡単な説明

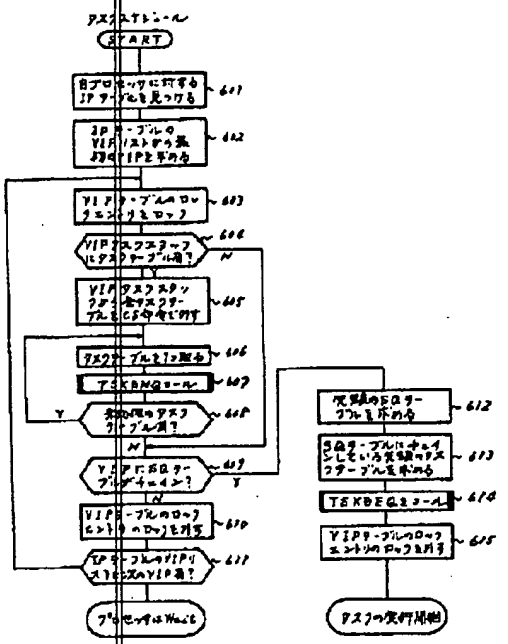
第１図は本発明の一実施例を示すタスクスケジュール処理の流れを示すフローチャート、第２図は本発明の一実施例のシステム構成を示す図、第３図はタスクテーブル、空間テーブル、ＳＱテーブルの関連を示す図、第４図はＩＰテーブルとＶＩＰテーブルの関連を示す図、第５図はＶＩＰとタスク、ＩＰとＶＩＰの関連表示例を示す図、第６図は従来方式によるシステム構成を示す図、第７図はタスクの実行可能化の処理の流れを示すフローチャート、第８図は負荷監視によるＩＰとＶＩＰの関連切替え処理の流れを示すフローチャート、第９図はＴＳＫＥＮＱ処理の流れを示すフローチャート、第１０図はＴＳＫＤＥＱ処理の流れを示すフローチャート、第１１図は本発明の他の実施例のシステム構成を示す図、第１２図はそのタスクスケジュール処理の流れを示すフローチャート、第１３図はタスクの実行可能化の処理の流れを示すフローチャート、第１４図はＳＱＥＮＱ処理の流れを示すフローチャート、第１５図はＡＳＳＣＨ処理の流れを示すフローチャート、第１６図は本発明の更に他の実施例のシステム構成

を示す図、第１７図は、本発明のさらに他の実施例のシステム構成を示す図、第１８図は、タスクを実行可能にするための処理の流れを示すフローチャート、第１９図は、タスクスケジュールの処理の流れを示すフローチャート、第２０図は、タスクデキュールチンの処理の流れを示すフローチャート、第２１図は、プロセッサ故障時の処理の流れを示すフローチャート、第２２図は、空間とＶＩＰタスクの関連表示例を示す図である。

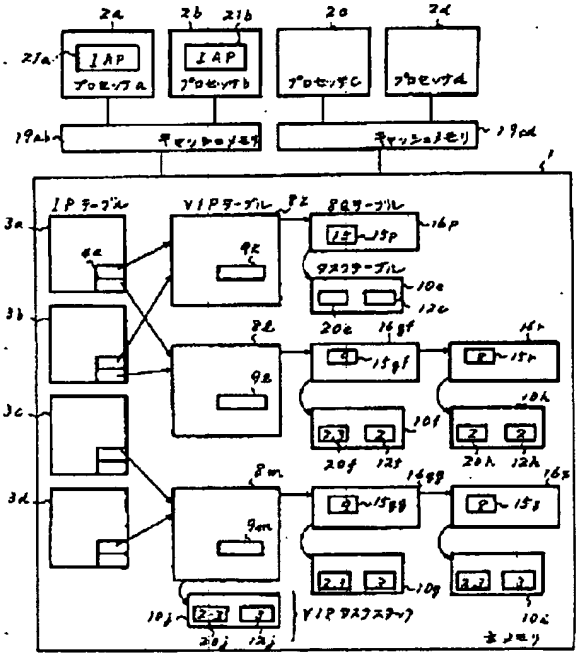
１：主メモリ、２ａ～２ｄ：プロセッサ、１８ａｂ，１８ｃｄ，１６１ａ～１６１ｄ：キャッシュメモリ、３ａ～３ｄ：ＩＰテーブル、８ａ～８ｄ，８ｋ～８ｍ：ＶＩＰテーブル、１０ｅ～１０ｊ：タスクテーブル、１６ｐ～１６ｓ：ＳＱテーブル、１４ｑ，１４ｒ，１４ｓ：空間テーブル、４ａ～４ｄ：サービス対象ＶＩＰビットマップ、１１０ｑ：ロックエントリ、１１２ｑ：仕掛有ＶＩＰビットマップ、１１３ｑ：ＶＩＰ対応カウンタ。

代理人弁理士　小　川　勝　男

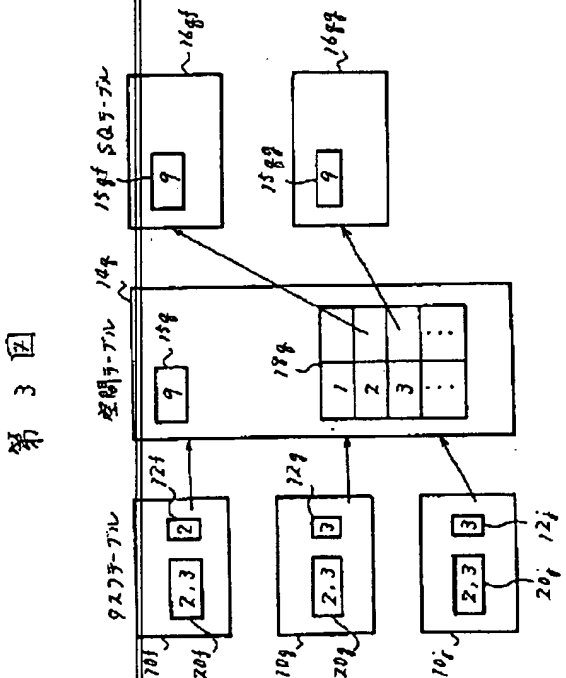
第１図



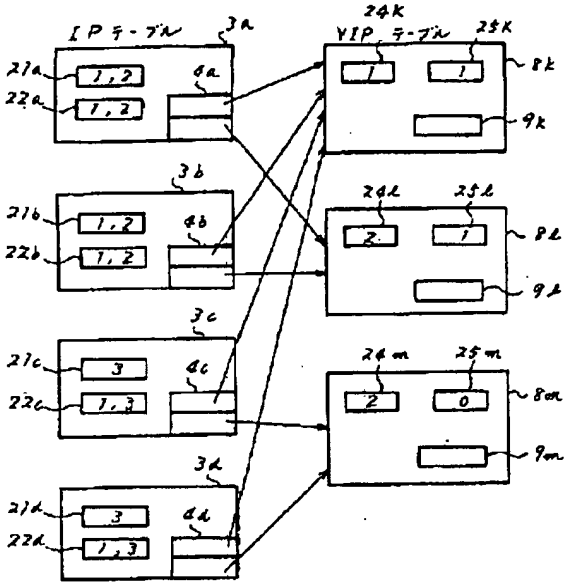
第２図



第３図



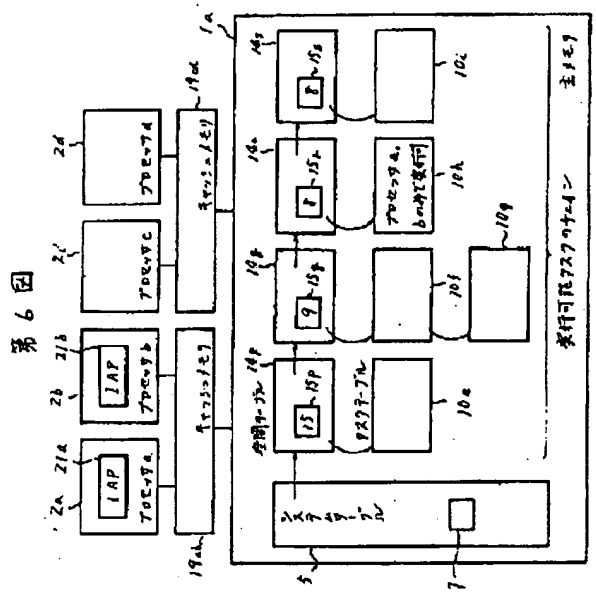
第 4 図



第 5 図

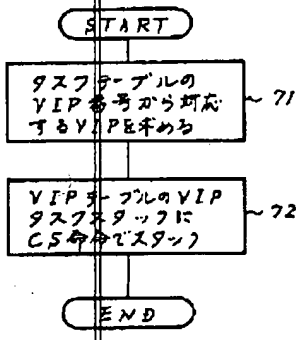
```
***VIP-TASK COUNT DISPLAY***
VIP#   TASK COUNT   AFFINITY TASK COUNT
 2         2              1
 3         2              0

***IP-VIP LIST DISPLAY***
 IP#   VIP LIST   HIGH LOAD   LOW LOAD
  1       12          12          12
  2       12          12          12
  3        3           3          13
  4        3           3          13
```

第 6 図

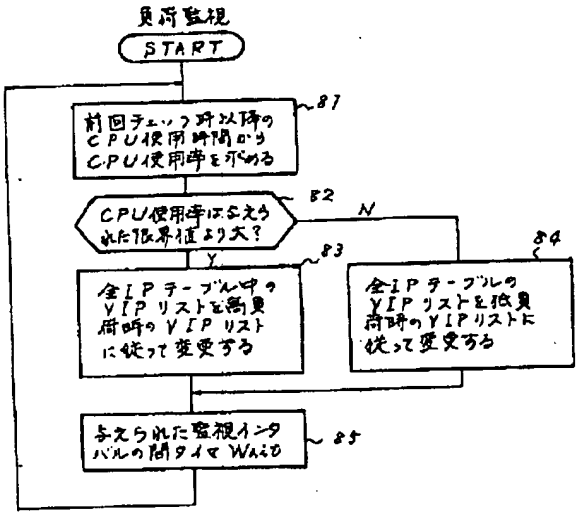


第 7 図

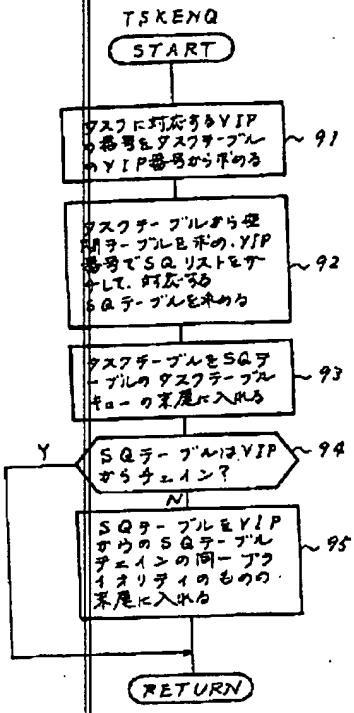
タスクを実行可能とした時の処理



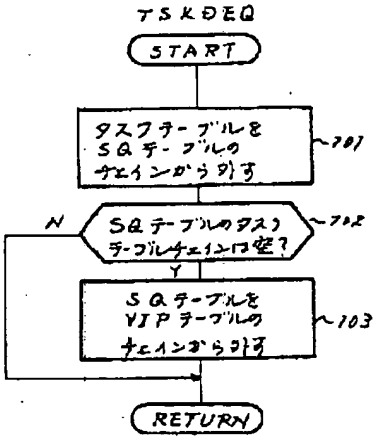
第 8 図



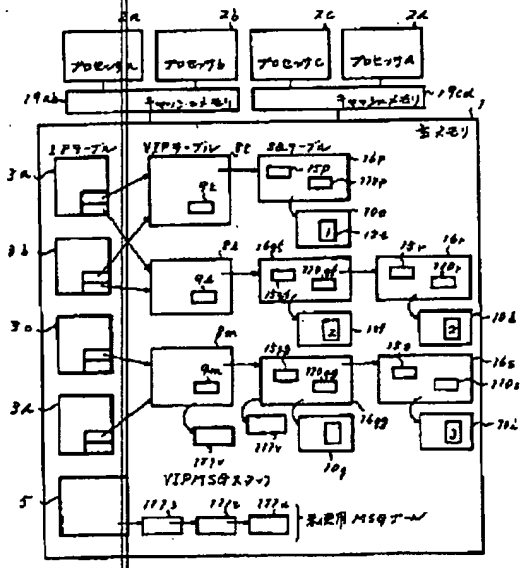
第 9 図



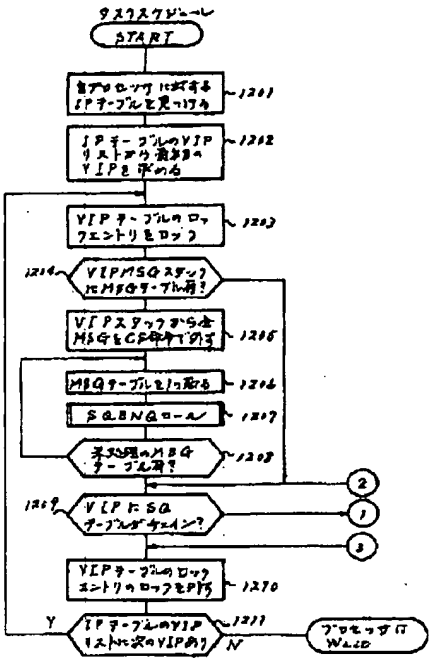
第 10 図



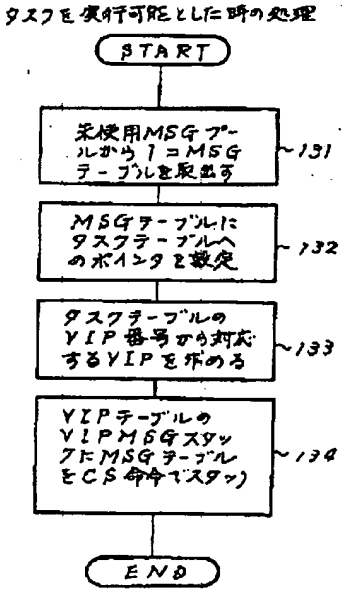
第 11 図



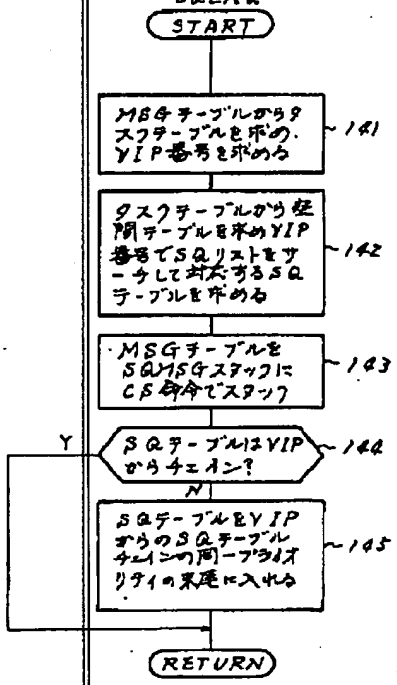
第12図(a)



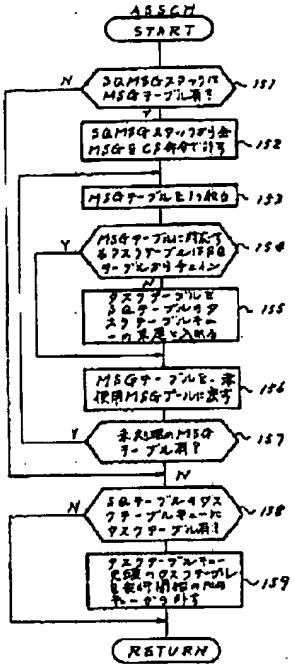
第13図



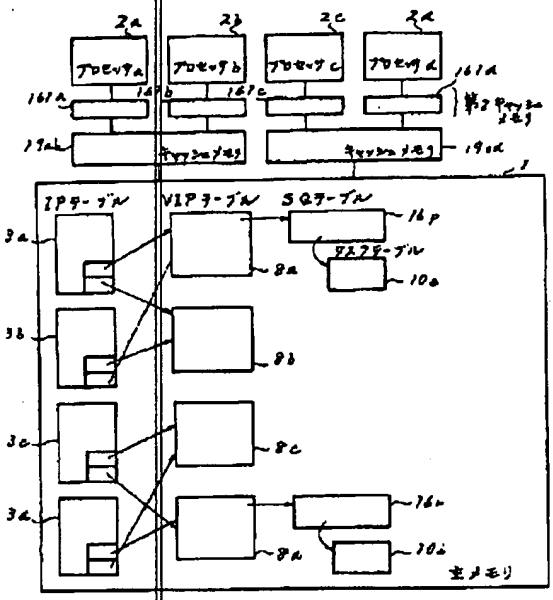
第14図



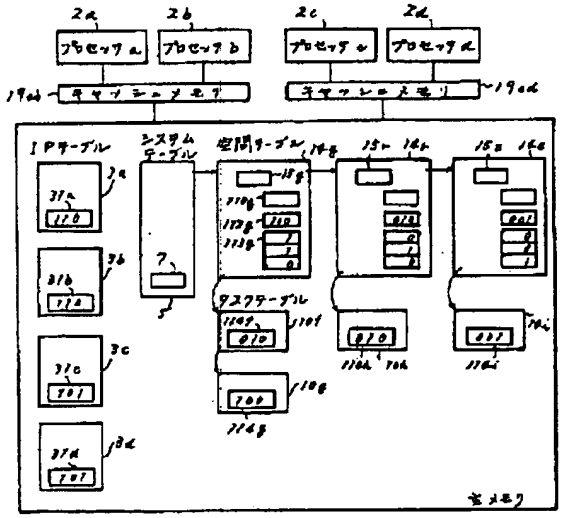
第15図



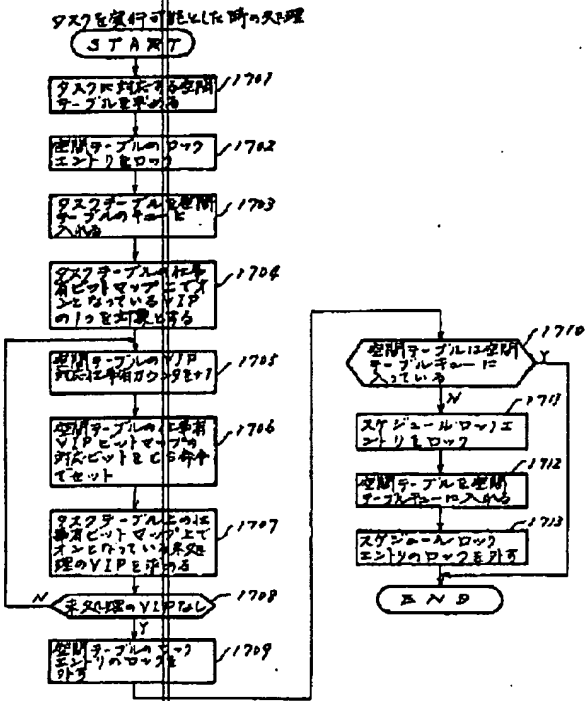
第16図



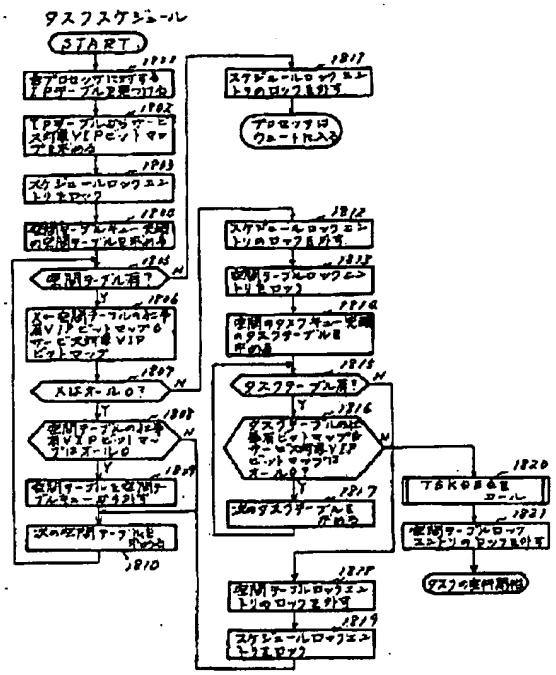
第17図



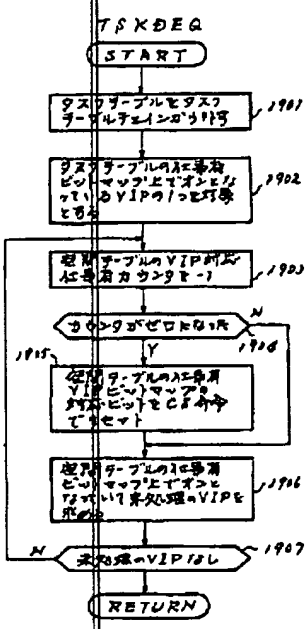
第18図



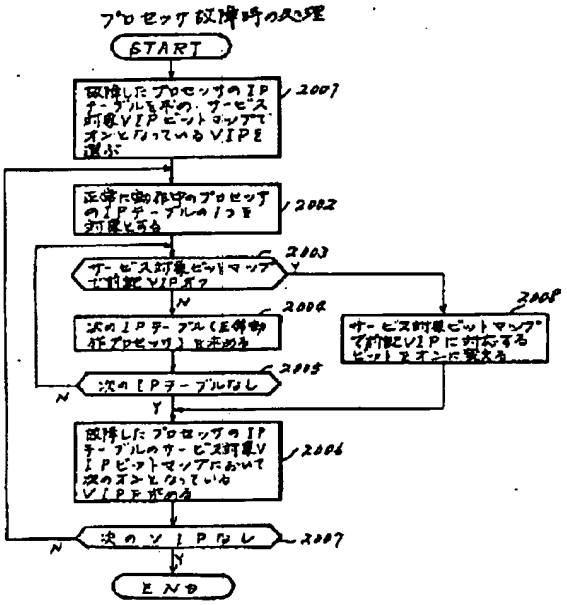
第19図



第20図



第21図



第22図

*** AS VIP-TASK COUNT DISPLAY ***

ASID	VIP0	VIP1	VIP2
1	1	1	0
2	0	1	0
3	0	0	1
4	0	0	0
TOTAL	1	2	1

第1頁の続き
優先権主張 ㉜平1(1989)11月21日㉝日本(JP)㉛特願 平1-304236

㉗発明者 佐藤 一浩 神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所ソフトウエア工場内

㉗発明者 原田 晃 神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所ソフトウエア工場内